One-vs-One classification for deep neural networks

Pawara, Pornntiwa; Okafor, Emmanuel; Groefsema, Marc; He, Sheng; Schomaker, Lambert R. B.; Wiering, Marco A.

Published in:
Pattern recognition

DOI:
10.1016/j.patcog.2020.107528

Publication date:
2020

Pattern Recognition 108 (2020) 107528

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

One-vs-One classification for deep neural networks

Pornntiwa Pawara[a,*], Emmanuel Okafor[b], Marc Groefsema[a], Sheng He[c], Lambert R.B. Schomaker[a], Marco A. Wiering[a]

[a] Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen, 9747 AG Groningen, The Netherlands
[b] Department of Computer Engineering, Ahmadu Bello University, Zaria, Nigeria
[c] Boston Children's Hospital, Harvard Medical School, USA

ARTICLE INFO

Article history:
Received 17 October 2019
Revised 19 June 2020
Accepted 30 June 2020
Available online 1 July 2020

Keywords:
Deep learning
Computer vision
Multi-class classification
One-vs-One classification
Plant recognition

ABSTRACT

For performing multi-class classification, deep neural networks almost always employ a One-vs-All (OvA) classification scheme with as many output units as there are classes in a dataset. The problem of this approach is that each output unit requires a complex decision boundary to separate examples from one class from all other examples. In this paper, we propose a novel One-vs-One (OvO) classification scheme for deep neural networks that trains each output unit to distinguish between a specific pair of classes. This method increases the number of output units compared to the One-vs-All classification scheme but makes learning correct decision boundaries much easier. In addition to changing the neural network architecture, we changed the loss function, created a code matrix to transform the one-hot encoding to a new label encoding, and changed the method for classifying examples. To analyze the advantages of the proposed method, we compared the One-vs-One and One-vs-All classification methods on three plant recognition datasets (including a novel dataset that we created) and a dataset with images of different monkey species using two deep architectures. The two deep convolutional neural network (CNN) architectures, Inception-V3 and ResNet-50, are trained from scratch or pre-trained weights. The results show that the One-vs-One classification method outperforms the One-vs-All method on all four datasets when training the CNNs from scratch. However, when using the two classification schemes for fine-tuning pre-trained CNNs, the One-vs-All method leads to the best performances, which is presumably because the CNNs had been pre-trained using the One-vs-All scheme.

1. Introduction

Convolutional neural networks (CNNs) have obtained excellent results for many different pattern recognition problems [1,2]. Most image recognition problems require the CNN to solve a multi-class classification problem. Whereas in the machine learning literature, different approaches have been proposed for dealing with multiple classes[3], in deep learning, the One-vs-All classification scheme is almost universally used. The problem of this method is that decision boundaries need to be learned that separate the examples of each class from examples of all other classes. Especially if images of different classes resemble each other quite a lot, learning such decision boundaries can be very complicated. Therefore, we propose a novel One-vs-One classification scheme for training CNNs in which each output unit only needs to learn to distinguish between examples of two different classes. This should make training the CNN easier and lead to better recognition performance.

Multi-class classification in machine learning. The best-known methods to deal with multi-class classification tasks are One-vs-All (OvA) classification and One-vs-One (OvO) classification [4]. Other approaches include One-class classification [5,6], hierarchical methods [7,8], and error-correcting output codes [9]. One-vs-All (OvA) classification is the most commonly used method for dealing with multi-class problems. In this classification scheme, multiple binary classifiers are trained to distinguish examples from one class from all other examples. When there are K classes, the OvA scheme trains K different classifiers. An advantage of this method is that machine learning algorithms that were designed for binary classification can be easily adapted in this way to deal with multi-class classification problems. A disadvantage is that the dataset on which each classifier is trained becomes imbalanced be-

* Corresponding author.
E-mail addresses: p.pawara@rug.nl (P. Pawara), m.a.wiering@rug.nl (M.A. Wiering).

https://doi.org/10.1016/j.patcog.2020.107528

cause there are many more negative examples than positive ones for each classifier.

The One-vs-One (OvO) classification method has also regularly been used for training particular machine learning algorithms such as support vector machines [10–12] or other classifiers [13]. In the OvO scheme, each binary classifier is trained to discriminate between examples of one class and examples belonging to one other class. Therefore, if there are K classes, the OvO scheme requires training and storing $K(K-1)/2$ different binary classifiers, which can be seen as a disadvantage when K is large. The authors in [14] described several methods to cope with a large set of base learners for OvO. Furthermore, different algorithms have been proposed to improve the OvO scheme [15,16]. An advantage of the OvO scheme is that the datasets of individual classifiers are balanced when the entire dataset is balanced. Comparisons between using the OvO scheme and the OvA scheme have shown that OvO is better for training support vector machines [10,17] and several other classifiers [13].

Multi-class classification in deep neural networks. When deep neural networks are used for multi-class classification problems, the output layer almost always uses a softmax function and one output unit for each different class. This is therefore a One-vs-All classification scheme, although the output units share the same hidden layers. Attribute learning [18,19], in which different attributes are predicted, and their combination is used to infer a class, is another promising way to deal with multi-class learning but may require substantially more labeling effort.

Contributions of this paper. We propose a novel One-vs-One classification method for deep neural networks. The proposed architecture comprises an output layer with $K(K-1)/2$ output units and a shared feature learning part. Each output is trained to distinguish between inputs of two classes and be indifferent to examples of other classes. To construct the OvO classification scheme, we devised three steps: 1) Creating a code matrix to transform the one-hot encoding to a new label encoding, 2) Changing the output layer and the loss function, and 3) Changing the method to classify new (test) examples.

This OvO scheme has to the best of our knowledge not been proposed before for deep neural networks. We only found one related paper that describes an OvO scheme for shallow neural networks, for which $K(K-1)/2$ different neural networks are trained and stored [20]. The advantages of our proposed OvO method compared to that more traditional OvO scheme are that we only need to train and store one deep neural network, and our architecture may benefit from positive knowledge transfer when training multiple output units together.

In our experiments, we use three different plant datasets (including a novel dataset called Tropic) and a dataset of different types of monkeys. Using computer vision techniques for classifying plant images plays a vital role in agriculture, monitoring the environment, and automatic plant detection systems [21]. Although much research has already been done on recognizing plant images, it is still a difficult and challenging task due to intra-class variations, inter-class similarities, and complex backgrounds [22,23].

We also use a different dataset consisting of types of monkeys to examine if the results on the plant recognition problems generalize to a different fine-grained species classification problem. Furthermore, we performed experiments with an imbalanced variant of the monkey dataset to study if the OvO scheme can better handle class imbalances. For classifying the image data, two deep CNNs are used, Inception-V3 [24] and ResNet-50 [25], which are trained from scratch or with fine-tuning from pre-trained weights. Finally, experiments were performed with different amounts of training images and classes from the four datasets using subsampling, to study the impact of smaller or larger datasets on the results obtained with the OvO and OvA schemes.

Paper Outline. The rest of this paper is organized as follows. Section 2 describes and theoretically compares the One-vs-One and One-vs-All classification methods for deep neural networks. Section 3 describes the plant datasets, the monkey dataset, and the data-augmentation methods. The experimental setup is presented in Section 4, after which Section 5 presents and discusses the results. Section 6 concludes the paper and describes directions for future work.

2. A Primer on One-vs-All and One-vs-One classification

In this section, we explain the two classification schemes (One-vs-All and One-vs-One) for multi-class classification with deep neural networks. Then, we present a theoretical analysis of the advantages of the One-vs-One scheme.

2.1. One-vs-All classification

In multi-class classification, each example belongs to precisely one class. Therefore a dataset is annotated with the correct class label using a one-hot target output vector containing zeros, except for the target class, which has a value of one. The goal is to learn a mapping between inputs and outputs so that the correct class obtains the highest activation and, preferably, is the only one that becomes activated after propagating the inputs to the outputs.

One-vs-All (OvA) classification involves training K different binary classifiers (output units), each designed to discriminate an instance of a given class relative to all other classes [26]. To do this, a softmax activation function is used in the output layer, and the weights of the deep neural network are optimized using the cross-entropy loss function and a particular optimizer.

The categorical cross-entropy loss J_{OvA} for a single training example is:

$$J_{OvA} = -\sum_{i=1}^{K} y_i \log(\hat{y}_i) \tag{1}$$

Where K denotes the number of classes, y_i is defined as the target value (0 or 1) for a given class i, and $\hat{y}_i$ denotes the probability assigned by the network that class i is the correct one. To compute these probabilities, the output values of the network are given to the softmax activation function:

$$\hat{y}_i = \frac{e^{o_i}}{\sum_{j=1}^{K} e^{o_j}} \tag{2}$$

Where o_i represents the output value for class i, which is computed by summing the weighted values passed from the final hidden layer. Note that this final summation uses a weight vector for each class and therefore the activations of the final hidden layer are linearly combined to compute the o_i values. For testing purposes on unseen examples, the predicted output class C is simply computed using:

$$C = \text{argmax}_i \hat{y}_i. \tag{3}$$

2.2. The proposed One-vs-One approach

In this subsection, we explain the novel One-vs-One (OvO) classification scheme for traning deep neural networks. As mentioned in the introduction, OvO classification has been used successfully for different machine learning algorithms such as support vector machines. This classification scheme has also been used for training neural networks [20], for which different (shallow) neural networks were trained separately for each pair of classes. Therefore, that approach leads to the necessity of training many neural networks and no possibility of sharing weights for solving multiple related pattern recognition problems. We present a novel OvO clas-

sification scheme that only requires to train a single (deep) neural network. This has as advantages that the method requires less storage space, computational time and can benefit from knowledge transfer and multi-task learning. To construct the OvO classification scheme, we devised three steps: 1) Creating a code matrix, 2) Changing the output layer and the loss function, and 3) Changing the method to classify new (test) examples. We will explain these steps in detail below.

Creating the OvO code matrix. In OvO classification, instead of using a one-hot target vector that assigns a one to the target class and zeros to all other classes, we need to construct a method that allows for pairwise classification. Therefore, instead of using K outputs where K is the number of classes, we need to construct a target vector consisting of $L = K(K-1)/2$ values. We do this by constructing a code matrix, which converts the one-hot target vector to the target values for the L outputs. The output units in the deep neural network represent binary classifiers with outputs in the bound [-1,1]. The target values for these outputs have values -1, 0, or 1. Here, the value 0 denotes that the output should be indifferent to both classes. For example, when an output unit needs to distinguish cats from dogs, and the training image shows a zebra, the target value for that output unit would be 0. The code matrix M_c has a dimension of $K \times L$. The arrangement of the code matrix entries uses the principle of pairwise separation of classes C_i and C_j, given that $i < j$ [4].

It is easiest to explain the code matrix using an example. Suppose we have a dataset with 5 classes, $K = 5$, so that the number of output units $L = (5 \times 4/2) = 10$. For this example, the code matrix is defined as:

$$M_c = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$$

When we have the one-hot target vector $\mathbf{y}$ denoting the correct class, we can multiply it with the code matrix to obtain the target outputs for the different output units. For example when $\mathbf{y^T} = (0\ 0\ 0\ 1\ 0)$, which denotes that class 4 is the correct one for a training example, then we can compute the target vector for OvO classification by: $\mathbf{y}^T_{ovo} = \mathbf{y^T} M_c = (0\ 0\ -1\ 0\ 0\ -1\ 0\ -1\ 0\ 1)$, which is simply a copy of the 4th row of the code matrix. In this example, the 3rd entry in the obtained target vector denotes that for the pairwise classification between classes 1 and 4, the target class is 4, so that the 3rd output unit should output a value of –1.

New output layer and loss function. As explained above, the OvO classification method requires more output units than OvA classification. Although this may mean the OvO scheme is complicated to use when there are a vast number of classes, many datasets do not have more than 50 classes, and in the experiments, we will focus on such (smaller) datasets. To allow the network to output pairwise classifications, we simply construct a deep model with $L = K(K-1)/2$ output units. We cannot use the softmax activation function anymore since that would assign probabilities to all output units, which add up to 1. Furthermore, the novel target output vector contains numbers between -1 and 1. Therefore, in our system, we use the hyperbolic tangent (tanh) activation function for the L output units, defined as:

$$\hat{y}_i = \frac{e^{o_i} - e^{-o_i}}{e^{o_i} + e^{-o_i}} \tag{4}$$

Although this network could be trained with the mean squared error (MSE) loss function, it is well-known that training a neural network for a classification problem can be better done with a cross-entropy loss function [27]. Therefore, we customized the binary cross-entropy loss function, for which the target values y_i^{OvO} and output values $\hat{y}_i$ are first scaled to the range [0,1] using:

$$y_i^{OvO'} = \frac{y_i^{OvO} + 1}{2}, \qquad y'_i = \frac{\hat{y}_i + 1}{2} \tag{5}$$

For dealing with numerical problems, the probability values of y' are clipped to lie in the range of [0.00001, 0.99999]. Now, the multi-output binary cross-entropy loss J_{OvO} for an example is computed with:

$$J_{OvO} = -\frac{1}{L}\sum_{i=1}^{L}(y_i^{OvO'} \times \log(y'_i) + (1 - y_i^{OvO'}) \times \log(1 - y'_i)) \tag{6}$$

Where $y_i^{OvO'}$ denotes the new target value for a given class i. Note that this loss function is also used for multi-label classification, where multiple outputs can be activated given an input pattern. The difference in our approach is that we include don't care target outputs as well, which need to be mapped to the probability 0.5 or a tanh-activation of 0 in the output layer to minimize the loss. Another choice would be to not train on such outputs at all, but that would provide less information to the network. Some preliminary experiments showed that better results were obtained by also training on target values of zero.

Classifying new examples. To predict the class label C for an input pattern $\mathbf{x}$, the input is first propagated to compute the L outputs $\hat{y}_i$. Then, a decoding scheme is used so that the votes of all binary OvO outputs are combined. For this, the same code matrix M_c is used to compute the summed class output vector $\mathbf{z}$ consisting of K elements:

$$\mathbf{z} = M_c\ \hat{\mathbf{y}}. \tag{7}$$

Note that this means that output vector should be similar to the corresponding values in the specific row in the code matrix, although don't care values are not important to get a large summed vote. Finally, the predicted class is selected by $C = \text{argmax}_i z_i$. The schematic representation for the deep neural network (InceptionV3) combined with the two classification methods is shown in Fig. 1(a) and Fig. 1(b).

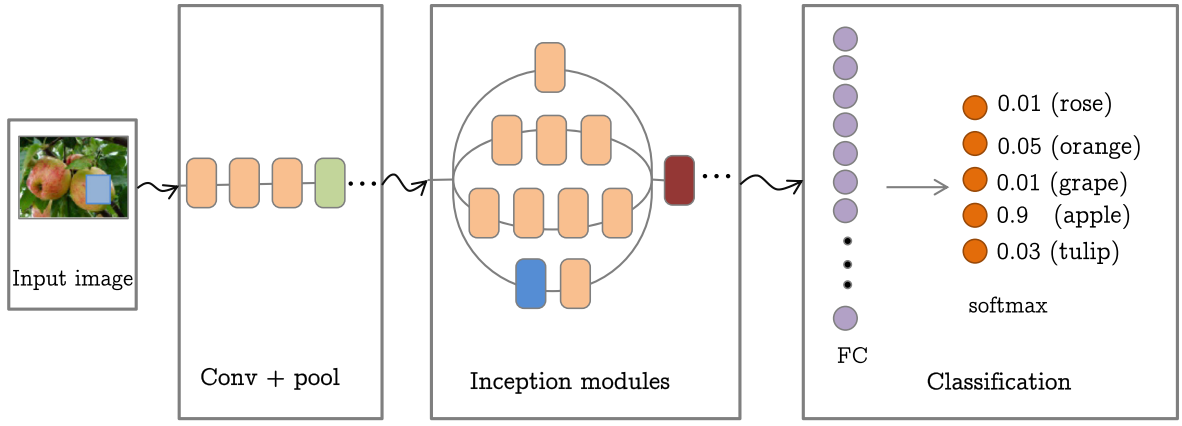
2.3. Analysis of the advantages of One-vs-One classification

In this subsection, we theoretically compare the One-vs-One and One-vs-All classification schemes. In our analysis, we will use simple binary classifiers for separating examples of one class from examples of one other class or examples of all other classes. Note that even in deep neural networks, the final output activations are usually computed using a weight matrix that connects the final hidden layer with each output unit. Therefore, the deep neural networks need to learn to map input patterns to linearly separable final hidden-layer activations. Each classifier first computes its output o_i using:

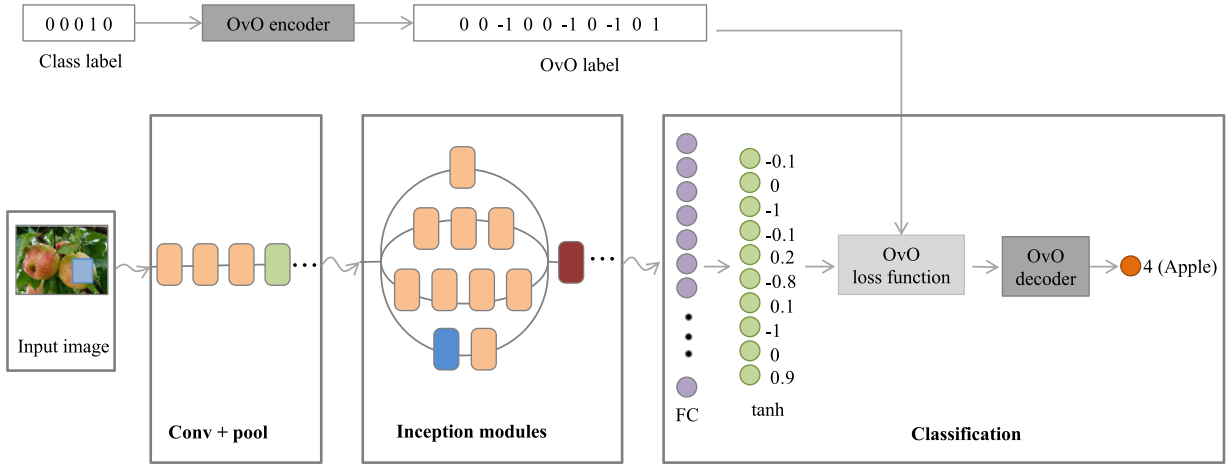
$$o_i = \mathbf{w_i^T} \cdot \mathbf{h} + b_i \tag{8}$$

Where b_i denotes the bias and $\mathbf{w_i}$ the weight vector for output i, and $\mathbf{h}$ denotes the vector containing all activations of the hidden units that are connected to the outputs. The OvA models use the softmax activation function to compute the class probabilities $\hat{y}_i = \frac{e^{o_i}}{\sum_j e^{o_j}}$ and the predicted class is given by $C = \text{argmax}_i \hat{y}_i$.

For simplicity reasons, in our analysis, the OvO models use a sigmoid activation function to discriminate between each pair of classes: $f_{ij} = \sigma(o_{ij})$, and we assume that $f_{ij} = 1 - f_{ji}$ for all $i \neq j$ and zero otherwise. Furthermore, we do not require these OvO models to output values close to 0.5 for different classes than the ones that are separated by the model. Note that the tanh activation function is a scaled sigmoid: $tanh(x) = 2\sigma(2x) - 1$, so this does



(a) One-vs-All



(b) One-vs-One

Fig. 1. The pipeline of the CNN showing a compact representation of Inception-V3 combined with the two classification systems; (a) One-vs-All (b) Multi-class One-vs-One. Note that the (...) represents several chains of neural network layers.

not impact our analysis. The predicted class for this OvO scheme on a test example is given by $C = \text{argmax}_i \sum_j f_{ij}$.

We assume a dataset $S = \{(\mathbf{x_1}, C_1), \ldots, (\mathbf{x_n}, C_n)\}$, where C_i denotes the number of the correct output class for input $\mathbf{x_i}$. First, we analyze if the OvO scheme is more powerful than the OvA scheme when separating different classes, for which we define multi-class separability for OvA and OvO.

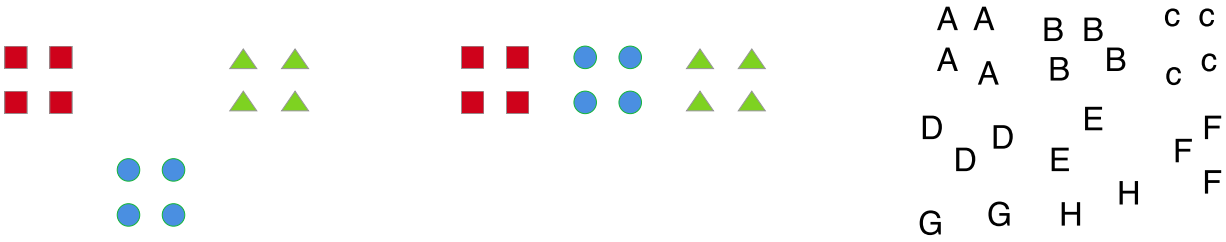
Definition: OvA separability. A mapping $\mathbf{h} = g(\mathbf{x}, \theta)$ separates all training examples with the OvA scheme, if there exist weight vectors $\mathbf{w_i}$ and biases b_i such that $\text{argmax}_i \hat{y}_i = \text{argmax}_i \mathbf{w_i^T h} + b_i = C$ for all $(\mathbf{x}, C) \in S$.

Definition: OvO separability. A mapping $\mathbf{h} = g(\mathbf{x}, \theta)$ separates all training examples with the OvO scheme, if there exist vectors $\mathbf{w_{ij}}$ and scalars b_{ij} s.t. $\text{argmax}_i \sum_j f_{ij} = \text{argmax}_i \sum_j \sigma(\mathbf{w_{ij}^T h} + b_{ij}) = C$ for all $(\mathbf{x}, C) \in S$.

We will first give an example with three linearly separable classes so that both the OvA and OvO scheme construct three decision boundaries, see Fig. 2(a). It should be clear that the three classes in Fig. 2(a) are linearly separable with OvA and OvO. The optimal decision boundaries are illustrated in Fig. 3(a) and Fig. 3(b).

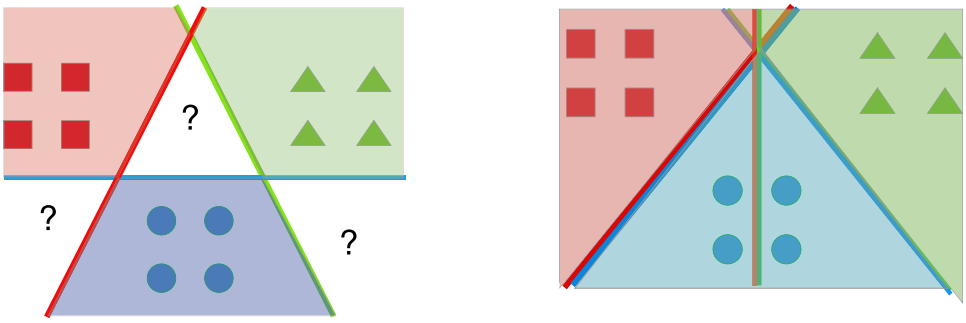
When we compare the decision boundaries for OvA and OvO, we observe several differences. First, the decision boundaries are placed in different ways. E.g., the red and green classes are separated by OvO by a vertical line in the middle. Second, with the OvO scheme, there is always one class that wins against all other classes for each input. For the OvA scheme, there are possible inputs for which there is no unique winner, such as points in the bottom left area where both the blue circle class and the red square class may have high outputs. The predicted class in such areas would depend on the exact weight vectors and bias values.

Now, let us examine the more complex problem shown in Fig. 2(b). The OvA scheme will have difficulties to learn to separate the blue circles from the examples of the other two classes. Although learning the correct decision boundaries is complicated for the OvA scheme, it is still possible. The blue-class model could have a higher bias value than the other models and be less sensitive to the input, and the other two classes could learn decision boundaries based on the x-axis. The OvO scheme can easily solve this problem, however, because linear divisions between each pair of classes are not hard to construct.



(a) Simple 3 classes. (b) Complex 3 classes. (c) Complex 8 classes.

Fig. 2. Three different multi-class problems of different complexities.



(a) Separation with OvA. (b) Separation with OvO.

Fig. 3. The optimal decision boundaries.

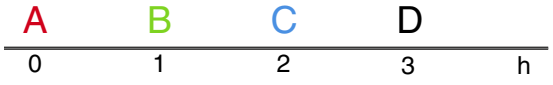


Fig. 4. 1D-Problem with 4 classes.

If we make the problem even more complex and add more classes, such as in Fig. 2(c), it seems impossible for the OvA scheme to separate all classes. However, also in this case the OvA scheme can linearly separate the classes, which we will prove below. It should be noted that it is much easier for the OvO scheme to handle such a dataset.

Now, suppose we have a dataset with K classes and one input dimension h, in which each class is linearly separable from each other class using the OvO scheme. Fig. 4 shows an example of such a problem with 4 classes A, B, C, and D. Note that for simplicity, we only drew a single data point for each class, but the analysis can be easily extended to multiple data points, as long as they lie close together. We now make the following proposition:

Proposition 1: If all pairs of classes are linearly separable (in one dimension), then the OvA scheme can also linearly separate all classes, but requires larger weight values to do this than the OvO scheme.

Proof of proposition 1: We assume we have K points $h_1, h_2, \ldots, h_K$ and K OvA models $f_i(h) = w_i h + b_i$. We require that each model f_i outputs the largest value on point h_i: $f_i(h_i) \geq f_j(h_i) + R$ for all $i, j \in \{1, 2, \ldots, K\}; i \neq j$. Here R is a positive constant that ensures the differences between model outputs are large enough so that the softmax function would output a value close to 1 for the winning class (e.g. $R = 3$).

It is not difficult to develop an algorithm that constructs the parameters w_i, b_i for all models f_i such that the above requirement holds. Let's look at the example of Fig. 4 again. In this example class A belongs to point $h = 0$, B to $h = 1$, C to $h = 2$, and D to $h = 3$. We have four models $f_z(h) = w_z h + b_z$, where z is the label (A, B, C, or D). For separating A and B, we require:

$$f_A(0) = f_B(0) + R \quad \text{and} \quad f_B(1) = f_A(1) + R. \tag{9}$$

There are multiple solutions, let's say we select:

$$f_A(h) = -Rh + 0.5R \quad \text{and} \quad f_B(h) = Rh - 0.5R. \tag{10}$$

It is easy to verify that the previous requirement is fulfilled with these two models. Now, for class C, we require:

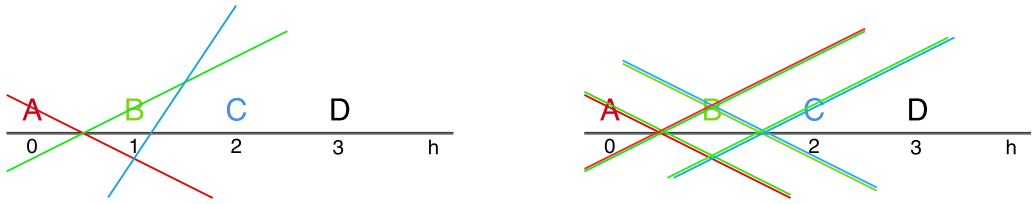
$$f_B(1) = f_C(1) + R \quad \text{and} \quad f_C(2) = f_B(2) + R. \tag{11}$$

From which follows: $f_C(h) = 3Rh - 3.5R$. When we continue this construction process, we also derive: $f_D(h) = 5Rh - 8.5R$.

We observe that the function $max_i f_i$ is piece-wise linear convex, which is illustrated for the models for A, B, and C in Fig. 5a.

It is easy to show that the algorithm can be generalized to multiple input dimensions. In the 1D case, we observed that the weights increase by $2R$ for each additional model, while the bias values become very negative. This finally leads to substantial weight values when there are many classes, and consequently, will decrease the generalization power. The weight-increase factor for each additional model depends on other problem-specific settings, such as the distance between examples in feature space δ (in our example $\delta = 1$), and the number of dimensions of the final hidden layer, H.

When dealing with H dimensions, the increase of the single weight can be spread over the H dimensions, so the increase of weights is $\frac{2R}{H}$ for each additional class. Therefore, projecting inputs to many hidden dimensions helps to have smaller weights, but many hidden units may also worsen generalization. When examples of different classes are closer together, the margin decreases, and the weight increase has to be multiplied with $\frac{1}{\delta}$. This also means that unbounded activation functions (e.g., ReLU) are useful for obtaining smaller weights in the final classification layer. When we take all these factors together, the OvA scheme's largest weight would be of the order $\frac{KR}{\delta H}$. E.g., for 50 classes ($K = 50$), $\delta = 0.1$,



(a) Solution of OvA. (b) Solution of OvO.

Fig. 5. The solutions for the 1D problem.

$R = 3$, and $H = 100$, the largest weights in the final classification layer could be around 15.

Now, examine how the OvO scheme solves the above problem. In this scheme, we use models of the form $f_{ij}(h) = w_{ij}h + b_{ij}$. For the first classes A and B, we require: $f_{AB}(0) = R$ and $f_{AB}(1) = -R$ to ensure that after applying the sigmoid function, the model incurs a small loss.

It is easy to see that for $f_{AB}(h)$ the weight w_{AB} equals $-2R$, similar to the OvA scheme. However, the different models do not depend on each other, and therefore the weights do not need to increase continuously. Furthermore, models that separate examples that are farther away from each other, such as $f_{AD}(h)$, can have much smaller weight values. The solution of the OvO scheme to the one-dimensional problem is illustrated in Fig. 5(b).

This concludes our proof of proposition 1. Both classification schemes can be used to separate the data projected to one dimension as long as examples of different classes lie close together, but the OvA model needs much larger weights if there are many classes. Another problem with the OvA scheme is that the different outputs heavily depend on each other. When one binary OvA classifier is adapted, other outputs have to be changed as well. Furthermore, when some outputs use large weight vectors in the final layer, their errors can have a significant impact on the training process. These two factors may increase instabilities of the training process.

The learned representation can indeed make up for the problems of the OvA scheme. For example, when the final hidden layer is very large, it is easier to learn decision boundaries with OvA. However, this could lead to strange generalization effects, as has also been shown in research on adversarial examples [27]. Furthermore, in the OvO scheme, outputs are affected by other outputs due to the shared feature-learning part, but this dependence also occurs for the OvA models. To conclude, the OvO scheme has the following advantages compared to the OvA scheme:

- The OvO scheme can have better generalization properties than the OvA scheme because there is less need for large weight vectors or a broad final feature representation, which is connected to the classification layer.
- In the OvA scheme, each binary classifier (output) is much more dependent on the other binary classifiers than in the OvO scheme, which could increase problems with learning instabilities.
- The OvO scheme does not introduce artificial class imbalances, whereas the OvA scheme does. If the dataset is balanced, the problem for each OvO classifier is balanced as well. For the OvA scheme, the dataset for each independent classifier is imbalanced.

Finally, we want to mention that although in general the OvO scheme requires training $K(K-1)/2$ different classifiers and therefore could cost much more training time than the OvA scheme, in our proposed architecture this is not the case. In the proposed OvO method, a single deep network is used that is trained on each example in the same way as in the OvA scheme. Only when there are very many classes (like thousands), the OvO scheme would become complex to store and train.

3. Datasets and data augmentation techniques

As mentioned in the introduction, plant image recognition systems have many applications. Convolutional neural networks (CNNs) have obtained remarkable results on different datasets for image-based plant classification [23,28–30]. In [31], two deep learning architectures, AlexNet and GoogLeNet, were trained on the PlantVillage dataset to detect plant leaves that contain diseases. The work described in [32] compared instances of Inception-V4, various instances of ResNet, and few other CNN models to classify diseases in plant images. Some works have also applied several other techniques to boost recognition performances, such as using different kinds of data augmentation [33,34] and transfer learning schemes [35].

In this section, we briefly describe the three different plant datasets, the monkey dataset, and the data augmentation methods used in our study.

3.1. Datasets

In this subsection, we describe the three plant datasets and the monkey dataset used in the experiments. Fig. 6 shows some example images from the plant datasets.

3.1.1. Agrilplant dataset

The AgrilPlant dataset was introduced in [36]. The dataset contains 3000 plant images with a uniformly distributed number of images per class. It contains 10 classes: Apple, Banana, Grape, Jackfruit, Orange, Papaya, Persimmon, Pineapple, Sunflower, and Tulip. Most of the images within this dataset contain variances in pose and object backgrounds. The dataset images were split in the proportion of 20% used for testing, and the remaining 80% of the images used for training.

3.1.2. Tropic dataset

The Tropic dataset contains 20 classes of plants with a total of 5276 images. Each of the classes contains a non-uniform distribution of images, varying from 221 to 371 images per class. The dataset contains the following plants: Acacia, Ashoka, Bamboo, Banyan, Chinese wormwood, Croton, Crown flower, Ervatamia, Golden shower, Hibiscus, Lady palm, Lime, Mango, Manila tamarind, Poinsettia, Raspberry ice Bougainvillea, Sanchezia, Umbrella tree, West Indian jasmine, and White plumeria. The images were collected by us during the day using a DSLR camera. The data was collected from diverse locations in Northeastern Thailand. All the images have similarities in illumination conditions but show different plant parts (flowers, branches, fruits, leaves, or the whole tree) and background information such as sky, houses, and soil. We randomly split the dataset in the ratio of 70% / 30% for the training and the testing set.

Fig. 6. Some example images from the three plant datasets for which we show one image per class for some classes in the datasets. The first row shows *AgrilPlant* images, the second row shows *Tropic* images, and the last row shows *Swedish* leaf images.

Fig. 7. Some example images from the *Monkey-10* dataset for which we show one image per class for all classes in the dataset.

3.1.3. Swedish dataset

The Swedish dataset [37] contains 1125 leaf images of 15 classes with 75 images per class. The leaf images were taken on a plain background. We adopted the same dataset splits as in previous studies using 25 randomly selected images per class for training and the rest of the images for testing.

3.1.4. Monkey-10 dataset

The Monkey-10 dataset[1] contains approximately 1400 images and 10 classes, and each class corresponds to a different species of monkeys. Each of the classes contains approximately 110 training images and 27 test images. The dataset consists of the following monkey species: Mantled howler, Patas monkey, Bald uakari, Japanese macaque, Pygmy marmoset, White-headed capuchin, Silvery marmoset, Common squirrel monkey, Black-headed night monkey, and Nilgiri langur. Fig. 7 shows some example images from the Monkey-10 dataset.

The Monkey-10 dataset was primarily used to observe if performance differences between the OvO and OvA schemes generalize to a different kind of fine-grained species dataset. Additionally, from the original Monkey-10 dataset, we randomly selected a non-uniform distribution of images from the training set, which varies from 10 to 120 images per class to create an imbalanced dataset. This dataset is called Imbalanced-Monkey-10 and serves as a purpose to study if the OvO or OvA scheme can better handle strongly imbalanced classes.

3.2. Data augmentation techniques

We applied three online data augmentation (DA) approaches during the training of the CNNs. The data-augmentation operations involve horizontal flipping, vertically shifting images up or down with random values with a maximum of 10% of the image height, and horizontally shifting images left or right with random values with a maximum of 10% of the image width (where novel pixels are filled in using nearest pixel values). These operation schemes were applied to all the training images of the datasets. The reason for using DA is to increase the size of the training dataset when training the CNN models.

4. Experimental setup

In this section, we present the different experimental setups in which we subsample the total amount of images and classes from the three plant datasets and the two monkey datasets. Afterwards we describe the experimental parameters used for training the two CNNs, Inception-V3 and ResNet-50.

4.1. Dataset sampling

This subsection describes two different forms of dataset sampling to obtain more dataset subsets that will be used in the experiments:

1. Dataset subsets with fewer classes: In the AgrilPlant dataset, we additionally considered 5 randomly selected classes from the original dataset; this version of the dataset is called AgrilPlant5 while the original dataset is called AgrilPlant10. For the Tropic dataset, we considered two additional subsets from the original dataset, which involves the random selection of 5 or 10 classes from the original dataset. Hence, we name the new and original datasets (Tropic5, Tropic10) and Tropic20, respectively. Similar considerations were made on the Swedish dataset for 5 and 10 randomly selected classes. Hence, this results in the new subset

[1] https://www.kaggle.com/slothkong/10-monkey-species.

Table 1
Number of training images per class after sub-sampling the datasets.

Train size	Dataset				
(%)	AgrilPlant	Tropic	Swedish	Monkey	Imbalanced-Monkey
10	24	15–26	2–3	10–12	1–12
20	48	31–52	5	21–24	2–24
50	120	77–130	12–13	52–61	5–61
80	192	124–207	20	84–98	8–98
100	240	155–259	25	105–120	10–120

variants; Swedish5 and Swedish10, while the original dataset is called Swedish15.
2. Dataset subsets in which the original training image examples (100%) were distributed into 10%, 20%, 50%, and 80% of the whole training set based on a random selection of the images. Table 1 shows the number of images per class of the datasets after sub-sampling. Note that the testing sets for the datasets were kept constant. Furthermore, we provide notations for describing the datasets using: < dataset name > < number of classes > ::ts < train size > . For example, Tropic20::ts10 denotes the Tropic dataset with 20 classes containing 10% of the training data.

The reason for performing experiments with the sub-sampling dataset variations is to determine how the CNN architectures combined with either the OvO or OvA classification system can deal with recognizing images under different conditions. The primary goal is to assess the performance variations of the two different classification schemes.

4.2. Deep CNN training schemes

Deep neural network architectures consist of several chains of neural network layers and operations: convolutional, normalization, non-linear activation functions, pooling, fully connected, and the final classification layer. In this study, we perform experiments with architectures which use inception modules (Inception-V3), and residual modules (ResNet-50). We chose these deep CNN architectures, because they are well known state-of-the-art architectures, but are based on different operations (inception or residual modules).

We trained the CNN models with two training schemes using the scratch or pre-trained version based on their use of random weights or pre-trained weights from the ImageNet dataset. Each of the training schemes employs the previously described deep convolutional neural networks (Inception-V3 and ResNet-50) combined with the OvA and OvO classification systems. The hyper-parameters were optimized using several preliminary experiments.

1. Scratch Experiments. The following experimental parameters were used: the previously described CNNs were initialized with random weights and trained for 200 epochs while optimizing the CNN loss function with the Adam optimizer, a batch size of 16, and a learning rate $l_r = 0.001$. The l_r decay uses a factor of 0.1 after every interval of 50 epochs. The scratch experiments on all the datasets were run within the computing time frame of $[10 - 130]$ minutes, depending on the given dataset/subset.
2. Fine-tuning Experiments. The following experimental parameters were used: the previously described CNNs were initialized with pre-trained weights from the ImageNet dataset. These models are trained for 100 epochs while optimizing the CNN loss function with the Adam optimizer, a batch size of 16, and a learning rate $l_r = 0.0001$. The l_r decay uses a factor of 0.1 after 50 epochs. The fine-tuning experiments on all the datasets were run within the computing time frame of $[6 - 66]$ minutes, depending on the given dataset/subset.

For all experiments, we used an NVIDIA V100 GPU with 28GB of memory.

5. Results and discussion

In this section, we present the classification performances of the two CNN methods (Inception-V3 and ResNet-50) combined with the two classification schemes (OvO and OvA) trained using the scratch or pre-trained instances of the CNN models on the three plant datasets, the monkey datasets, and some of the plant datasets without data augmentation on the training sets.

5.1. Results of scratch-Inception-V3

We trained the scratch Inception-V3 CNN based on five-fold cross-validation. The results obtained during the testing phase are reported in Table 2.

1. Evaluation of the CNN on the AgrilPlant Dataset: from Table 2a, we observe that training Scratch-Inception-V3 (CNN) combined with OvO significantly outperforms the CNN combined with OvA ($p < 0.05$) on 3 dataset subsets with a smaller training size. Another observation is that the CNN combined with OvO surpasses the CNN combined with OvA on the AgrilPlant5::ts10 dataset with a significant difference of ~ 5.5%.
2. Evaluation of the CNN on the Tropic Dataset: from Table 2(b), we observe that training Scratch-Inception-V3 combined with OvO significantly outperforms the CNN combined with OvA ($p < 0.05$) on 6 dataset subsets.
3. Evaluation of the CNN on the Swedish Dataset: from Table 2(c), we observe that training the CNN combined with OvO significantly outperforms the CNN combined with OvA ($p < 0.05$) on 8 datasets (subsets or whole). Another observation is that the CNN combined with OvO surpasses the CNN combined with OvA on the Swedish10::ts10 dataset with a significant difference of 8.5%.

5.2. Results of scratch-ResNet-50

We trained the scratch ResNet-50 combined with the two classification schemes using five-fold cross-validation. The results obtained during the testing phase are reported in Table 3.

1. Evaluation of the CNN on the AgrilPlant Dataset: from Table 3(a), we observe that training Scratch-ResNet-50 combined with OvO significantly outperforms the CNN combined with OvA on 4 smaller subsets.
2. Evaluation of the CNN on the Tropic Dataset: from Table 3(b), we observe that training the CNN combined with OvO significantly outperforms the CNN combined with OvA on 6 subsets of this dataset. Another observation is that the CNN combined with OvO surpasses the CNN combined with OvA on the Tropic10::ts{10,20} subsets with a significant difference of ~ 5%.

Table 2
Recognition performances (average accuracy and standard deviation) of Scratch-Inception-V3 combined with the two classification methods. The bold numbers indicate significant differences between the classification methods ($p < 0.05$).

(a) The AgrilPlant dataset

Train size	AgrilPlant5		AgrilPlant10	
(%)	OvO	OvA	OvO	OvA
10	**77.13 ± 1.28**	71.67 ± 2.67	**77.80** ± 3.00	73.57 ± 1.47
20	85.47 ± 2.10	83.33 ± 3.47	86.97 ± 1.69	85.87 ± 1.57
50	**92.40 ± 0.86**	89.73 ± 1.19	94.87 ± 1.00	94.57 ± 1.23
80	94.47 ± 0.90	94.33 ± 0.53	96.47 ± 0.69	96.60 ± 0.73
100	94.93 ± 0.37	94.80 ± 1.02	96.90 ± 0.65	97.40 ± 0.67

(b)The Tropic dataset

Train size	Tropic5		Tropic10		Tropic20	
(%)	OvO	OvA	OvO	OvA	OvO	OvA
10	**82.24 ± 1.91**	78.76 ± 2.09	**75.14** ± 2.73	70.46 ± 3.22	66.51 ± 4.72	65.93 ± 3.31
20	89.06 ± 1.55	89.40 ± 1.47	**86.77** ± 1.14	83.43 ± 2.06	81.48 ± 4.52	80.57 ± 1.35
50	**97.19 ± 0.66**	95.74 ± 1.15	95.59 ± 1.28	94.78 ± 0.34	94.62 ± 1.67	94.47 ± 0.46
80	**98.84 ± 0.53**	98.02 ± 0.47	98.38 ± 0.70	97.42 ± 0.73	97.87 ± 0.34	**97.21 ± 0.31**
100	99.13 ± 0.51	98.30 ± 1.06	98.56 ± 0.46	98.54 ± 0.22	98.18 ± 0.96	98.03 ± 0.14

(c)The Swedish dataset

Train size	Swedish5		Swedish10		Swedish15	
(%)	OvO	OvA	OvO	OvA	OvO	OvA
10	**71.60 ± 4.24**	66.08 ± 3.01	**79.52 ± 3.43**	70.96 ± 4.19	**72.91 ± 5.29**	65.41 ± 3.32
20	86.40 ± 2.61	86.96 ± 4.36	**91.84 ± 2.25**	85.60 ± 3.90	**88.73 ± 1.98**	84.99 ± 2.71
50	**98.40 ± 0.75**	95.36 ± 2.63	97.36 ± 0.86	97.36 ± 0.96	95.71 ± 1.41	94.99 ± 1.85
80	**99.36 ± 0.36**	98.56 ± 0.61	99.20 ± 0.58	98.48 ± 0.39	98.19 ± 0.49	97.41 ± 0.75
100	99.76 ± 0.36	99.44 ± 0.67	99.48 ± 0.18	99.00 ± 0.51	**98.59 ± 0.28**	97.76 ± 0.45

Table 3
Recognition performances (average accuracy and standard deviation) of Scratch-ResNet-50 combined with the two classification methods. The bold numbers indicate significant differences between the classification methods ($p <$ .05).

(a) The AgrilPlant dataset

Train size	AgrilPlant5		AgrilPlant10	
(%)	OvO	OvA	OvO	OvA
10	**77.53 ± 0.96**	72.93 ± 3.85	**76.23 ± 2.06**	72.93 ± 2.04
20	**85.40 ± 0.64**	82.73 ± 2.29	86.03 ± 1.29	84.20 ± 1.91
50	**91.47 ± 0.90**	89.87 ± 0.77	93.13 ± 0.46	93.20 ± 0.83
80	93.53 ± 1.22	93.73 ± 1.50	96.00 ± 0.53	95.03 ± 1.19
100	94.33 ± 0.94	93.87 ± 2.06	96.10 ± 0.38	96.23 ± 0.85

(b) The Tropic dataset

Train size	Tropic5		Tropic10		Tropic20	
(%)	OvO	OvA	OvO	OvA	OvO	OvA
10	**77.31 ± 1.05**	73.59 ± 2.63	**67.57 ± 3.44**	62.38 ± 1.42	59.78 ± 2.05	59.59 ± 2.27
20	87.41 ± 3.72	83.35 ± 3.45	**82.57 ± 1.75**	77.85 ± 2.10	**79.79 ± 0.72**	76.61 ± 1.31
50	93.47 ± 2.48	91.19 ± 2.40	93.45 ± 1.20	93.09 ± 0.76	93.31 ± 0.61	93.11 ± 1.02
80	97.29 ± 1.35	96.23 ± 0.89	96.45 ± 1.20	96.43 ± 0.88	96.49 ± 0.48	95.70 ± 0.70
100	98.64 ± 0.82	97.48 ± 0.44	97.44 ± 0.42	97.10 ± 0.57	**97.59 ± 0.23**	96.80 ± 0.43

(c) The Swedish dataset

Train size	Swedish5		Swedish10		Swedish15	
(%)	OvO	OvA	OvO	OvA	OvO	OvA
10	**75.20 ± 1.96**	71.76 ± 1.95	**73.52 ± 3.57**	63.44 ± 1.99	66.11 ± 4.18	66.83 ± 2.49
20	86.80 ± 3.26	83.53 ± 1.61	82.32 ± 4.81	83.60 ± 2.53	84.05 ± 4.12	82.21 ± 1.81
50	96.08 ± 0.95	96.48 ± 1.34	95.56 ± 0.83	95.68 ± 0.99	93.31 ± 0.90	93.15 ± 1.20
80	98.24 ± 0.83	97.92 ± 0.91	**98.00 ± 0.40**	97.12 ± 0.46	96.19 ± 1.00	96.03 ± 0.61
100	98.96 ± 0.46	98.72 ± 0.52	98.40 ± 0.37	98.32 ± 0.23	**97.28 ± 0.35**	96.24 ± 0.94

3. Evaluation of the CNN on the Swedish Dataset: from Table 3(c), we observe that training the CNN combined with OvO significantly outperforms the CNN combined with OvA on 4 subsets of this dataset. Furthermore, the CNN combined with OvO surpasses the CNN combined with OvA on the Swedish10::ts10 dataset with a difference of ~ 10%.

5.3. Results of fine-tuned inception-V3

We trained the pre-trained Inception-V3 based on five-fold cross-validation. The results obtained during the testing phase are shown in Table 4.

Table 4
Recognition performances (average accuracy and standard deviation) of Fine-tuned-Inception-V3 combined with the two classification methods. The bold numbers indicate significant differences between the classification methods ($p < .05$).

(a) The AgrilPlant dataset

Train size	AgrilPlant5		AgrilPlant10	
(%)	OvO	OvA	OvO	OvA
10	88.67 ± 2.13	90.40 ± 2.42	92.13 ± 1.52	**94.87 ± 0.88**
20	92.27 ± 2.09	92.07 ± 1.86	94.47 ± 1.77	**96.67 ± 0.59**
50	96.20 ± 1.66	96.27 ± 1.14	97.13 ± 1.02	98.03 ± 0.77
80	96.27 ± 1.16	97.53 ± 0.69	97.93 ± 0.51	**98.77 ± 0.57**
100	97.00 ± 1.18	97.07 ± 1.23	98.07 ± 0.56	98.83 ± 0.53

(b) The Tropic dataset

Train size	Tropic5		Tropic10		Tropic20	
(%)	OvO	OvA	OvO	OvA	OvO	OvA
10	97.15 ± 1.72	96.61 ± 2.50	92.93 ± 1.21	94.60 ± 1.52	90.42 ± 2.88	**93.60 ± 0.94**
20	97.39 ± 1.22	98.74 ± 0.99	96.01 ± 0.98	**98.25 ± 0.57**	95.70 ± 0.36	**96.67 ± 0.52**
50	99.32 ± 0.32	99.47 ± 0.56	98.75 ± 0.27	**99.53 ± 0.41**	98.43 ± 0.21	**99.20 ± 0.10**
80	99.66 ± 0.13	99.61 ± 0.22	99.32 ± 0.23	**99.79 ± 0.15**	99.05 ± 0.35	99.46 ± 0.23
100	99.76 ± 0.24	99.81 ± 0.32	99.56 ± 0.22	**99.87 ± 0.16**	99.33 ± 0.09	**99.68 ± 0.12**

(c) The Swedish dataset

Train size	Swedish5		Swedish10		Swedish15	
(%)	OvO	OvA	OvO	OvA	OvO	OvA
10	94.88 ± 4.10	92.48 ± 4.23	84.56 ± 2.56	**91.72 ± 4.44**	87.52 ± 4.78	86.11 ± 2.04
20	97.44 ± 3.26	97.52 ± 3.06	97.68 ± 1.40	98.96 ± 0.71	95.55 ± 2.34	94.48 ± 3.33
50	99.68 ± 0.18	**99.98 ± 0.04**	99.72 ± 0.11	99.84 ± 0.17	99.23 ± 0.40	99.20 ± 0.21
80	99.92 ± 0.18	99.92 ± 0.18	99.76 ± 0.17	99.88 ± 0.11	99.60 ± 0.27	99.81 ± 0.20
100	99.92 ± 0.18	99.92 ± 0.18	99.92 ± 0.11	99.92 ± 0.18	99.79 ± 0.15	**99.97 ± 0.06**

Table 5
Recognition performances (average accuracy and standard deviation) of Fine-tuned ResNet-50 combined with the two classification methods. The bold numbers indicate significant differences between the classification methods ($p < .05$).

(a) The AgrilPlant dataset

Train size	AgrilPlant5		AgrilPlant10	
(%)	OvO	OvA	OvO	OvA
10	91.13 ± 1.39	89.47 ± 3.03	93.13 ± 1.57	93.17 ± 0.31
20	93.93 ± 2.47	92.40 ± 1.16	95.83 ± 1.87	96.17 ± 0.87
50	96.33 ± 1.62	96.07 ± 0.64	97.73 ± 1.11	97.67 ± 0.94
80	97.27 ± 0.86	97.07 ± 1.34	98.40 ± 0.48	98.47 ± 0.40
100	97.60 ± 1.44	97.33 ± 1.33	98.47 ± 0.70	98.63 ± 0.70

(b) The Tropic dataset

Train size	Tropic5		Tropic10		Tropic20	
(%)	OvO	OvA	OvO	OvA	OvO	OvA
10	96.80 ± 1.45	96.61 ± 1.20	92.54 ± 1.91	91.96 ± 1.20	90.54 ± 1.09	90.76 ± 1.40
20	98.16 ± 0.88	97.87 ± 1.09	95.80 ± 0.89	**97.70 ± 0.30**	93.96 ± 0.49	**96.27 ± 0.42**
50	99.52 ± 0.38	99.22 ± 0.47	98.72 ± 0.29	**99.19 ± 0.17**	98.17 ± 0.63	**99.05 ± 0.10**
80	99.66 ± 0.37	99.56 ± 0.32	99.24 ± 0.28	**99.71 ± 0.25**	98.80 ± 0.21	**99.38 ± 0.15**
100	99.66 ± 0.28	99.76 ± 0.24	99.58 ± 0.11	99.71 ± 0.17	99.23 ± 0.18	**99.49 ± 0.16**

(c) The Swedish dataset

Train size	Swedish5		Swedish10		Swedish15	
(%)	OvO	OvA	OvO	OvA	OvO	OvA
10	90.48 ± 4.79	89.68 ± 6.14	90.40 ± 2.37	87.88 ± 1.88	84.32 ± 4.39	85.47 ± 3.22
20	97.44 ± 1.85	98.08 ± 2.14	98.76 ± 0.96	96.80 ± 2.04	97.47 ± 2.54	94.32 ± 3.62
50	99.76 ± 0.36	99.60 ± 0.28	99.60 ± 0.20	99.72 ± 0.23	99.47 ± 0.27	99.49 ± 0.33
80	99.76 ± 0.36	99.92 ± 0.18	99.92 ± 0.18	99.68 ± 0.39	99.71 ± 0.17	99.79 ± 0.24
100	99.92 ± 0.18	99.92 ± 0.18	99.92 ± 0.11	99.92 ± 0.18	99.65 ± 0.49	99.68 ± 0.20

1. Evaluation of the CNN on the AgrilPlant Dataset: from Table 4(a), the results show that there are 3 subsets of this dataset where training the Fine-tuned-Inception-V3 combined with OvA significantly outperforms the CNN combined with OvO.
2. Evaluation of the CNN on the Tropic Dataset: from Table 4(b), we observe that the CNN combined with OvA significantly outperforms the CNN combined with OvO on 8 subsets of the Tropic10 and Tropic20 datasets.
3. Evaluation of the CNN on the Swedish Dataset: from Table 4(c), we observe that training the CNN combined with OvA significantly outperforms the CNN combined with OvO on 3 subsets of this dataset. Another observation is that the CNN combined with OvA surpasses the CNN combined with OvO

on the Swedish10::ts10 dataset with a significant difference of ~ 7%.

5.4. Results of fine-tuned ResNet-50

We trained the pre-trained ResNet-50 combined with the two classification methods based on five-fold cross-validation. The results obtained during the testing phase are reported in Table 5.

1. Evaluation of the CNN on the AgrilPlant Dataset: from Table 5(a), we observe that training the CNN combined with OvO results in similar performance levels to the CNN combined with OvA on this dataset.
2. Evaluation of the CNN on the Tropic Dataset: from Table 5(b), we observe that training the CNN combined with OvA significantly outperforms the CNN combined with OvO on 7 subsets of the datasets with more classes.
3. Evaluation of the CNN on the Swedish Dataset: from Table 5(c), the results show that there is no significant difference between training the CNN with the two classification methods on all subsets of this dataset.

5.5. Results on the monkey datasets

We trained the two CNNs from scratch or using pre-trained weights using the two classification methods on the two monkey datasets, Monkey-10 and Imbalanced-Monkey-10, based on five-fold cross-validation. The results obtained during the testing phase are reported in Table 6.

1. Evaluation of Scratch Inception-V3 on the Monkey-10 and Imbalanced-Monkey-10 datasets: from Table 6(a), we observe that training the CNN combined with OvO significantly outperforms the CNN combined with OvA on 5 (smaller) subsets of the Monkey-10 datasets with several times significant differences of ~ 7%.
2. Evaluation of Scratch Resnet-50 on the Monkey-10 and Imbalanced-Monkey-10 datasets: from Table 6(b), we observe that training the CNN combined with OvO on Monkey-10 results in one case in a significantly better performance (Monkey10:ts10) with a significant difference of 5%.
3. Evaluation of Fine-tuned Inception-V3 on the Monkey-10 and Imbalanced-Monkey-10 datasets: from Table 6(c), we observe that training the CNN combined with OvA significantly outperforms the CNN combined with OvO on one data subset of Monkey-10 and Imbalanced-Monkey-10.
4. Evaluation of Fine-tuned Resnet-50 on the Monkey-10 and Imbalanced-Monkey-10 datasets: from Table 6(d), the results show that there is no significant difference between training the CNN with the two classification methods on both the Monkey-10 and the Imbalanced-Monkey-10 dataset.

5.6. Results of training CNNs without data augmentation

We trained the two CNNs from scratch and using pre-trained weights combined with the two classification methods on the Agril5::ts100 and Tropic10::ts100 datasets without data augmentation on the training data (again based on five-fold cross-validation). The results obtained during the testing phase are reported in Table 7.

The results show that training Scratch-ResNet-50 combined with OvO significantly outperforms the CNN combined with OvA on the AgrilPlant5::ts100 dataset with a significant difference of ~ 4%. Another observation is that the CNNs combined with OvO always perform a bit better than the CNNs combined with OvA on these two datasets. When we compare these results to the results when data augmentation is used, we can observe that data augmentation leads to performance improvements between 3% and 13%. We also note that especially Scratch-ResNet-50 profits a lot from data augmentation.

Table 6
Recognition performances (average accuracy and standard deviation) of the studied CNNs combined with the two classification methods applied on the Monkey-10 datasets. The bold numbers indicate significant differences between the classification methods ($p < .05$).

(a) Scratch Inception-V3

Train size	Monkey10		Imbalanced-Monkey10	
(%)	OvO	OvA	OvO	OvA
10	**55.91 ± 1.12**	48.68 ± 5.35	38.11 ± 3.38	35.04 ± 3.49
20	**68.91 ± 2.45**	61.47 ± 3.70	**48.24 ± 4.90**	41.17 ± 4.78
50	**86.28 ± 0.63**	84.10 ± 1.95	**66.79 ± 1.99**	61.97 ± 2.63
80	93.00 ± 1.73	90.94 ± 1.94	75.33 ± 1.67	72.04 ± 3.31
100	94.16 ± 1.70	92.69 ± 1.19	78.25 ± 1.78	75.99 ± 2.34

(b) Scratch Resnet-50

Train size	Monkey10		Imbalanced-Monkey10	
(%)	OvO	OvA	OvO	OvA
10	**54.52 ± 2.49**	49.49 ± 0.98	36.43 ± 4.20	34.39 ± 2.41
20	67.66 ± 3.48	62.91 ± 3.27	42.57 ± 5.79	40.64 ± 3.43
50	80.81 ± 2.83	81.46 ± 1.19	63.64 ± 3.00	59.55 ± 3.10
80	89.56 ± 2.07	89.64 ± 0.71	70.22 ± 3.89	68.32 ± 2.77
100	92.33 ± 1.41	90.73 ± 1.30	74.53 ± 2.47	72.47 ± 3.39

(c) Fine-tuned Inception-V3

Train size	Monkey10		Imbalanced-Monkey10	
(%)	OvO	OvA	OvO	OvA
10	95.69 ± 1.42	96.86 ± 1.32	78.85 ± 6.24	75.11 ± 2.67
20	97.44 ± 1.07	97.15 ± 2.03	84.32 ± 3.27	84.46 ± 4.81
50	97.52 ± 0.73	98.17 ± 0.94	93.22 ± 2.61	94.66 ± 2.07
80	97.67 ± 1.15	**99.13 ± 0.41**	93.86 ± 1.88	**96.57 ± 1.39**
100	98.76 ± 0.66	99.27 ± 0.52	94.66 ± 2.49	96.42 ± 1.61

(d) Fine-tuned Resnet-50

Train size	Monkey10		Imbalanced-Monkey10	
(%)	OvO	OvA	OvO	OvA
10	92.40 ± 1.75	91.61 ± 1.35	64.15 ± 2.95	63.93 ± 2.96
20	94.53 ± 1.53	94.37 ± 2.24	79.85 ± 1.68	74.17 ± 5.89
50	95.77 ± 0.97	96.79 ± 1.65	89.70 ± 2.44	85.41 ± 4.48
80	97.37 ± 0.64	97.37 ± 1.40	92.55 ± 2.06	91.61 ± 2.67
100	97.66 ± 1.36	97.96 ± 0.48	93.86 ± 1.79	91.69 ± 1.73

5.7. Discussion

We now summarize all obtained results when data augmentation is used:

- When training the two CNNs from scratch, the OvO classification method performs significantly better in 37 out of the 100 experiments. In this case, the OvA method never significantly outperforms the OvO method.
- When training the two pre-trained CNNs by fine-tuning them on the four datasets, the OvA method performs significantly better in 23 out of the 100 experiments. In this case, the OvO method never significantly outperforms the OvA method.
- The improvements of OvO when the CNNs are trained from scratch are larger for smaller datasets. When we examine dataset subsets of 10%, 20%, and 50%, the OvO scheme performs significantly better in 29 out of 60 experiments. This agrees with the theory stating that the OvO scheme generalizes better than the OvA scheme.

We also observed that the training process is generally more stable with the OvO method than with the OvA scheme. In Fig. 8, we show two train and test loss curves on a small dataset when

Table 7
Recognition performances (average accuracy and standard deviation) of the studied CNNs combined with the two classification methods applied on the Agril5::ts100 and Tropic10::ts100 datasets. The bold number indicates a significant difference between the classification methods ($p < .05$).

Models	AgrilPlant5::ts100		Tropic10::ts100	
	OvO	OvA	OvO	OvA
Scratch-Inception-V3	91.47 ± 1.73	89.33 ± 4.43	94.15 ± 4.28	91.84 ± 5.51
Scratch-Resnet50	**87.60 ± 1.57**	83.53 ± 1.80	84.89 ± 0.87	84.40 ± 1.82
Fine-tuned-Inception-V3	93.40 ± 1.64	92.53 ± 2.60	96.50 ± 0.88	95.20 ± 5.04
Fine-tuned-Resnet50	92.53 ± 0.61	91.80 ± 1.79	93.74 ± 1.18	93.53 ± 1.31

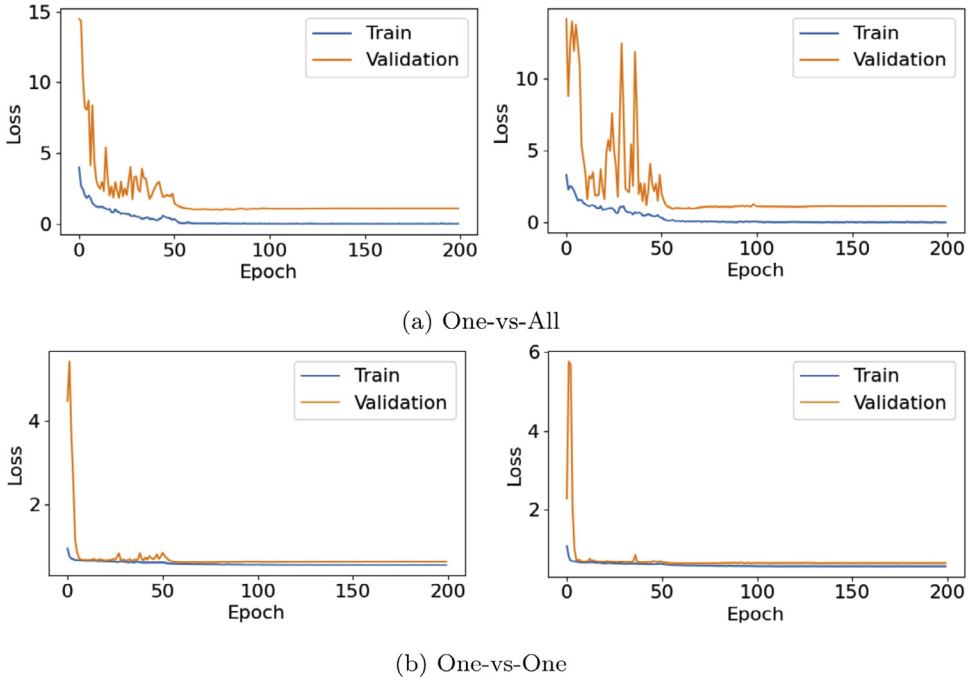


Fig. 8. Two loss curves when training Scratch-ResNet-50 combined with the classification methods on the AgrilPlant10::ts10 dataset; (a) One-vs-All, and (b) One-vs-One.

training ResNet-50 from scratch. The plots clearly show a more stable learning process for OvO, which agrees with the theory that it is beneficial to have output units which are not heavily dependent on each other.

We finally want to mention several last points, which we noticed by analyzing all results. First, the results of using pre-trained weights are typically better than the results of training the architectures from scratch. This holds for both classification methods, but the differences are much larger for the OvA scheme. Second, the performances of Inception-V3 are overall a bit better than the results of ResNet-50. The best results on the original datasets are excellent and were obtained with the pre-trained Inception-V3 architecture combined with the OvA scheme. The best performance on the AgrilPlant10 dataset is 98.8% (see Table 4(a)). The best performance on the Tropic20 dataset is 99.7% (see Table 4(b)). The best result on the Swedish15 dataset is 99.97% (see Table 4(c)). The best result on the Monkey-10 dataset is 99.3% (see Table 6(c)).

6. Conclusion

We described a novel technique for training deep neural networks based on the One-vs-One classification scheme. Two convolutional neural network architectures were trained using the One-vs-One scheme and the standard One-vs-All scheme on four image datasets with different amounts of examples and classes. The results show that when the deep neural networks are trained from scratch, the proposed method significantly outperforms the conventional One-vs-All training scheme in 37 out of 100 experiments. The results also show that this is not the case when the architectures were fine-tuned, for which the One-vs-All scheme wins in 21 out of 100 experiments. A possible reason why the OvA training scheme performs better with fine-tuning is that the architectures were pre-trained using the One-vs-All scheme on ImageNet. It would be interesting to train One-vs-One architectures on ImageNet and study if this would improve the transfer learning results.

Future work. There are several directions that we want to explore further. First, instead of using the One-vs-One scheme, it would be interesting to generalize our method to the use of error-correcting output codes [9]. The proposed architecture can also be extended by connecting the One-vs-One outputs to an additional One-vs-All output layer.

Second, although transfer learning is very useful for solving a different image recognition problem, there are also quite different applications involving fMRI images, 3D medical scans, or hyperspectral camera-images. For such pattern recognition problems, almost no pre-trained architectures exist. We would therefore like to research the benefits of using One-vs-One classification for such problems.

Third, we want to study the benefits of using One-vs-One classification when combined with other deep neural networks, such as recurrent neural networks (RNNs). The training process of recurrent neural networks is usually much less stable than when training convolutional neural networks, and it would be interesting to study if the One-vs-One scheme is beneficial for training RNNs.

Declaration of Competing Interest

The authors declare that they have known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Peregrine high performance computing cluster.

References

[1] J. Schmidhuber, Deep learning in neural networks: an overview, Neural networks 61 (2015) 85–117.
[2] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436.
[3] M. Aly, Survey on multiclass classification methods, Neural networks 19 (2005) 1–9.
[4] E. Alpaydin, Introduction to machine learning, The MIT Press, 2014.
[5] D.M.J. Tax, One-class classification: Concept learning in the absence of counter-examples, Technische Universiteit Delft, 2001 Ph.D. thesis.
[6] Tao Ban, S. Abe, Implementing multi-class classifiers by one-class classification methods, in: The 2006 IEEE International Joint Conference on Neural Network Proceedings, 2006, pp. 327–332.
[7] S. Kumar, J. Ghosh, M.M. Crawford, Hierarchical fusion of multiple classifiers for hyperspectral data analysis, Pattern Analysis and Applications 5 (2002) 210–220.
[8] V. Vural, J.G. Dy, A hierarchical method for multi-class support vector machines, in: Proceedings of the Twenty-First International Conference on Machine Learning, 2004, pp. 105–113.
[9] T.G. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes, Journal of Artificial Intelligence Research 2 (1) (1995) 263–286.
[10] E.L. Allwein, R.E. Schapire, Y. Singer, Reducing multiclass to binary: a unifying approach for margin classifiers, Journal of Machine Learning Research 1 (2001) 113141.
[11] M. Galar, A. Fernández, E. Barrenechea, F. Herrera, DRCW-OVO: distance-based relative competence weighting combination for one-vs-one strategy in multi–class problems, Pattern Recognit 48 (1) (2015) 28–42.
[12] Z.-L. Zhang, X.-G. Luo, S. García, J.-F. Tang, F. Herrera, Exploring the effectiveness of dynamic ensemble selection in the one-versus-one scheme, Knowl Based Syst 125 (2017) 53–63.
[13] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes, Pattern Recognit 44 (8) (2011) 17611776.
[14] A. Rocha, S.K. Goldenstein, Multiclass from binary: expanding one-versus-all, one-versus-one and ECOC-based approaches, IEEE Trans Neural Netw Learn Syst 25 (2) (2014) 289–302.
[15] Y. Liu, J.-W. Bi, Z.-P. Fan, A method for multi-class sentiment classification based on an improved One-vs-One (OVO) strategy and the support vector machine (SVM) algorithm, Inf Sci (Ny) 394 (2017) 38–52.
[16] P. Songsiri, V. Cherkassky, B. Kijsirikul, Universum selection for boosting the performance of multiclass support vector machines based on one-versus-one strategy, Knowl Based Syst 159 (2018) 9–19.
[17] C.W. Hsu, C.J. Lin, A comparison of methods for multiclass support vector machines, IEEE Trans. Neural Networks 13 (2) (2002) 415–425.
[18] A. Farhadi, I. Endres, D. Hoiem, D. Forsyth, Describing objects by their attributes, in: Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 1778–1785.
[19] S. He, L. Schomaker, Open set Chinese character recognition using multi-typed attributes, arXiv preprint arXiv:1808.08993 (2018).
[20] G. Ou, Y.L. Murphey, Multi-class pattern classification using neural networks, Pattern Recognit 40 (1) (2007) 418.
[21] X. Wang, J. Liang, F. Guo, Feature extraction algorithm based on dual-scale decomposition and local binary descriptors for plant leaf recognition, Digit Signal Process 34 (2014) 101–107.
[22] D. Guru, Y. Sharath, S. Manjunath, Texture features and KNN in classification of flower images, IJCA, Special Issue on RTIPPR (1) (2010) 21–29.
[23] A. Fuentes, S. Yoon, S.C. Kim, D.S. Park, A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition, Sensors 17 (9) (2017) 2022.
[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.
[25] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[26] R. Rifkin, A. Klautau, In defense of one-vs-all classification, Journal of machine learning research 5 (Jan) (2004) 101–141.
[27] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.
[28] J.R. Ubbens, I. Stavness, Deep plant phenomics: a deep learning platform for complex plant phenotyping tasks, Front Plant Sci 8 (2017) 1190.
[29] A.C. Cruz, A. Luvisi, L. De Bellis, Y. Ampatzidis, X-Fido: an effective application for detecting olive quick decline syndrome with deep learning and data fusion, Front Plant Sci 8 (2017) 1741.
[30] J. Ubbens, M. Cieslak, P. Prusinkiewicz, I. Stavness, The use of plant models in deep learning: an application to leaf counting in rosette plants, Plant Methods 14 (1) (2018) 6.
[31] S.P. Mohanty, D.P. Hughes, M. Salathé, Using deep learning for image-based plant disease detection, Front Plant Sci 7 (2016) 1419.
[32] E.C. Too, L. Yujian, S. Njuki, L. Yingchun, A comparative study of fine-tuning deep learning models for plant disease identification, Comput. Electron. Agric. 161 (2019) 272–279.
[33] C. Zhang, P. Zhou, C. Li, L. Liu, A convolutional neural network for leaves recognition using data augmentation, in: Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, 2015 IEEE International Conference, 2015, pp. 2143–2150.
[34] P. Pawara, E. Okafor, L. Schomaker, M. Wiering, Data augmentation for plant classification, in: International Conference on Advanced Concepts for Intelligent Vision Systems, Springer, 2017, pp. 615–626.
[35] C. Douarre, R. Schielein, C. Frindel, S. Gerth, D. Rousseau, Transfer learning from synthetic data applied to soil–root segmentation in x-ray tomography images, Journal of Imaging 4 (5) (2018) 65.
[36] P. Pawara, E. Okafor, O. Surinta, L. Schomaker, M. Wiering, Comparing local descriptors and bags of visual words to deep convolutional neural networks for plant recognition., in: ICPRAM, 2017, pp. 479–486.
[37] O. Söderkvist, Computer vision classification of leaves from Swedish trees, Linköping University, 2001 Master's thesis.

Pornntiwa Pawara is a Ph.D. student in Artificial Intelligence, the University of Groningen, the Netherlands. She received a masters degree in Computer Science from the University of Wollongong, Australia. Her research interests include computer vision, deep learning, and artificial intelligence.

Emmanuel Okafor earned a Ph.D. degree in Artificial Intelligence from the University of Groningen, the Netherlands,in 2019. Dr. Okafor is a lecturer in the Department of Computer Engineering, Ahmadu Bello University, Nigeria. His main research interests include computer vision, deep learning, control systems, reinforcement learning, robotics, and optimization.

Marc Groefsema is currently finishing his masters degree in Artificial Intelligence at the University of Groningen. He received his bachelor degree in AI in 2016. Besides studying he is an active assistant in the robotics laboratory. His research interests include cognitive robotics, image processing and machine learning.

Sheng He gained a cum laude Ph.D. degree in artificial intelligence from the University of Groningen, the Netherlands, in 2017. In 2018, he joined Harvard Medical School as a research fellow. He received the Chinese government award for outstanding self-financed students abroad (2016) from the Chinese Scholarship Council.

Lambert Schomaker is a Professor in Artificial Intelligence at the University of Groningen and was the director of its AI institute ALICE from 2001 to 2018. Prof. Schomaker is a senior member of IEEE and currently a chair of the data science and systems complexity center (DSSC) at FSE.

Marco Wiering is an assistant professor in the department of artificial intelligence from the University of Groningen. Dr. Wiering has (co-)authored more than 160 conference or journal papers. His main research topics are reinforcement learning, deep learning, neural networks, support vector machines, computer vision and optimization.